AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application.

LISTING OF CLAIMS:

1. (Currently amended) A method of queue management, said method comprising the steps of:

dividing a buffer memory into a plurality of memory segments, each memory segment comprising a plurality of bytes;

constructing a plurality of queues, wherein ~~each~~ at least one of said queues is assembled from two ~~one~~ or more memory segments;

providing a write pointer and a read pointer for each queue; and

providing a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue.

2. (Original) The method according to claim 1, further comprising the steps of:

incrementing said write pointer when data is written to a queue;

allocating an available memory segment when the current memory segment becomes full; and

setting the next segment pointer associated with the current memory segment to point to the memory segment allocated to the queue.

3. (Currently amended) The method according to claim 1, further comprising the step of providing a segment status table wherein an ~~said~~ entry in said segment status table is adapted to indicate whether a corresponding memory segment is occupied or available for use in a queue.

4. (Original) The method according to claim 1, further comprising the step of flushing each queue separately.

5. (Original) The method according to claim 1, further comprising the step of flushing all queues simultaneously.

6. (Currently amended) A method of queue management, said method comprising the steps of:

dividing a buffer memory into a plurality of memory segments, each memory segment comprising a plurality of bytes;

constructing a plurality of queues, wherein each queue is assembled from one or more memory segments;

providing a write pointer and a read pointer for each queue;

providing a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue; and

~~The method according to claim 1, further comprising the step of~~ flushing a queue wherein the memory segments comprising a queue are released up to but not including the memory segment corresponding to said write pointer.

7. (Original) The method according to claim 6, wherein said step of flushing comprises indicating that a memory segment is free via a segment status bit associated with each memory segment released.

8. (Original) The method according to claim 1, further comprising the steps of:

incrementing said read pointer when data is read from a queue; and

setting said read pointer in accordance with the next segment pointer associated with the memory segment currently being read when the end of a memory segment is reached.

9. (Currently amended) A method of queue management, said method comprising the steps of:

dividing a buffer memory into a plurality of memory segments, each memory segment comprising a plurality of bytes;

constructing a plurality of queues, wherein each queue is assembled from one or more memory segments;

providing a write pointer and a read pointer for each queue;

providing a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue; ~~The method according to claim 1, further comprising the steps of:~~

providing an initial write pointer for each queue established;

incrementing said write pointer when data is written to a queue;

holding said initial write pointer constant while data is written to a queue; and

setting said initial write pointer to the value of the write pointer when an end of packet is detected.

10. (Currently amended) A method of queue management, said method comprising the steps of:

dividing a buffer memory into a plurality of memory segments, each memory segment comprising a plurality of bytes;

constructing a plurality of queues, wherein each queue is assembled from one or more memory segments;

providing a write pointer and a read pointer for each queue;

providing a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue; ~~The method according to claim 1, further comprising the steps of:~~

providing an initial read pointer and a final read pointer for each queue established;

incrementing said read pointer when data is read from a queue;

holding said initial read pointer and said final read pointer constant while data is read from a queue;

setting said final read pointer to the value of said read pointer and said read pointer to the value of said initial read pointer when an end of packet is detected;

if an acknowledgement is received, setting said initial read pointer and said final read pointer to the value of said read pointer; and

if an acknowledgement is not received, re-reading data from said queue from said initial read pointer through said final read pointer.

11. (Original) The method according to claim 10, further comprising the step of releasing the memory segments in said queue between said initial read pointer and said final read pointer.

12. (Original) The method according to claim 1, further comprising providing an indication of the current size of each queue.

13. (Original) The method according to claim 1, further comprising providing an indication of the combined total size of all queues.

14. (Original) The method according to claim 1, further comprising the step of indicating when the size of a particular queue exceeds a user defined threshold.

15. (Original) The method according to claim 1, where in the number of memory segments and next segment pointers equals 64.

16. (Original) The method according to claim 1, wherein each memory segment comprises 64 bytes.

17. (Original) The method according to claim 1, wherein the number of queues equals 9.

18. (Currently amended) A queue management system, comprising:

a buffer memory divided into a plurality of memory segments, each memory segment comprising a plurality of bytes;

means for constructing a plurality of queues, wherein ~~each~~ at least one of said queues is assembled from ~~one~~ two or more memory segments;

a write pointer and a read pointer associated with each queue; and

a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue.

19. (Original) The system according to claim 18, further comprising:

means for incrementing said write pointer when data is written to a queue;

means for allocating an available memory segment when the current memory segment becomes full; and

means for setting the next segment pointer associated with the current memory segment to point to the memory segment allocated to the queue.

20. (Currently amended) The system according to claim 18, further comprising a segment status table wherein ~~said~~ an entry in said segment status table is adapted to indicate whether a corresponding memory segment is occupied or available for use in a queue.

21. (Original) The system according to claim 18, further comprising means for flushing each queue separately.

22. (Original) The system according to claim 18, further comprising means for flushing all queues simultaneously.

23. (Currently amended) A queue management system, comprising:

a buffer memory divided into a plurality of memory segments, each memory segment comprising a plurality of bytes;

means for constructing a plurality of queues, wherein each queue is assembled from one or more memory segments;

a write pointer and a read pointer associated with each queue;

a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue; and

~~The system according to claim 18, further comprising~~ means for flushing a queue wherein the memory segments comprising a queue are released up to but not including the memory segment corresponding to said write pointer.

24. (Original) The system according to claim 23, wherein said means for flushing is adapted to indicate that a memory segment is free via a segment status bit associated with each memory segment released.

25. (Original) The system according to claim 18, further comprising:

means for incrementing said read pointer when data is read from a queue; and

means for setting said read pointer in accordance with the next segment pointer associated with the memory segment currently being read when the end of a memory segment is reached.

26. (Currently amended) A queue management system, comprising:

a buffer memory divided into a plurality of memory segments, each memory segment comprising a plurality of bytes;

means for constructing a plurality of queues, wherein each queue is assembled from one or more memory segments;

a write pointer and a read pointer associated with each queue;

a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue;

~~The system according to claim 18, further comprising:~~ an initial write pointer associated with each queue established;

means for incrementing said write pointer while data is written to a queue;

means for holding said initial write pointer constant while data is written to a queue; and

means for setting said initial write pointer to the value of the write pointer when an end of packet is detected.

27. (Currently amended) A queue management system, comprising:

a buffer memory divided into a plurality of memory segments, each memory segment comprising a plurality of bytes;

means for constructing a plurality of queues, wherein each queue is assembled from one or more memory segments;

a write pointer and a read pointer associated with each queue;

a plurality of next segment pointers, each next segment pointer associated with a different memory segment and adapted to indicate the next memory segment in a queue;

~~The system according to claim 18, further comprising:~~ an initial read pointer and a final read pointer associated with each queue established;

means for incrementing said read pointer when data is read from a queue;

means for holding said initial read pointer and said final read pointer constant while data is read from a queue;

means for setting said final read pointer to the value of said read pointer and said read pointer to the value of said initial read pointer when a end of packet is detected;

means for setting said initial read pointer and said final read pointer to the value of said read pointer if acknowledgement is received; and

means for re-reading data from said queue from said initial read pointer through said final read pointer if an acknowledgment is not received.

28. (Original) The system according to claim 27, further comprising means for releasing the memory segments in said queue between said initial read pointer and said final read pointer.

29. (Original) The system according to claim 18, further comprising indicating means adapted to indicate the current size of each queue.

30. (Original) The system according to claim 18, further comprising indicating means adapted to indicate the combined total size of all queues.

31. (Original) The system according to claim 18, further comprising means for indicating when the size of a particular queue exceeds a user defined threshold.

32. (Original) The system according to claim 18, wherein the number of memory segments and next segment pointers equals 64.

33. (Original) The system according to claim 18, wherein each memory segment comprises 64 bytes.

34. (Original) The system according to claim 18, wherein the number of queues equals 9.

35. (Original) The system according to claim 18, wherein said buffer memory comprises a dual ported memory.

36. (Original) A dynamic queuing system, comprising:

a buffer memory divided into a plurality of memory segments, each memory segment comprising a plurality of bytes;

a segment controller operative to construct a plurality of queues, wherein each queue is assembled from one or more of said memory segments, said segment controller adapted to maintain a plurality of next segment pointers and segment status bits, each next segment pointer associated with a memory segment and adapted to indicate the next memory segment in a queue, each segment status bit indicating the availability of a corresponding memory segment;

write circuitry adapted to maintain a separate write pointer associated with each queue, said write circuitry adapted to write data to the appropriate memory segment associated with a particular queue; and

read circuitry adapted to maintain a separate read pointer associated with each queue, said read circuitry adapted to read data from the appropriate memory segment associated with a particular queue.

37. (Original) The system according to claim 36, wherein said write circuitry comprises:

means for incrementing said write pointer when data is written to a queue;

means for allocating an available memory segment when the current memory segment becomes full; and

means for setting the next segment pointer associated with the current memory segment to point to the memory segment allocated to the queue.

38. (Original) The system according to claim 36, further comprising means for flushing a queue wherein the memory segments comprising a queue are released up to but not including the memory segment corresponding to said write pointer.

39. (Original) The system according to claim 36, further comprising means for flushing a queue wherein the segment status bits associated with the memory segments making up a queue are cleared up to but not including the memory segment corresponding to said write pointer.

40. (Original) The system according to claim 36, wherein said read circuitry comprises:

means for incrementing said read pointer when data is read from a queue; and

means for setting said read pointer in accordance with the next segment pointer associated with the memory segment currently being read when the end of a memory segment is reached.

41. (Original) The system according to claim 36, wherein said write circuitry comprises:

an initial write pointer associated with each queue established;

means for incrementing said write pointer while data is written to a queue;

means for holding said initial write pointer constant while data is written to a queue; and

means for setting said initial write pointer to the value of the write pointer when an end of packet is detected.

42. (Original) The system according to claim 36, wherein said read circuitry comprises:

an initial read pointer and a final read pointer associated with each queue established;

means for incrementing said read pointer when data is read from a queue;

means for holding said initial read pointer and said final read pointer constant while data is read from a queue;

means for setting said final read pointer to the value of said read pointer and said read pointer to the value of said initial read pointer when an end of packet is detected;

means for setting said initial read pointer rand said final read pointer to the value of said read pointer if an acknowledgement is received; and

means for re-reading data from said queue from said initial read pointer through said final read pointer if an acknowledgement is not received.

43. (Original) The system according to claim 42, further comprising means for releasing the memory segments in said queue between said initial read pointer and said final read pointer.